Brief Announcement: Reliable End-User Communication Under a Changing Packet Network Protocol*

Brendan Juba†
MIT CSAIL & Harvard SEAS
Cambridge, MA, USA
bjuba@alum.mit.edu

ABSTRACT

We present the first end-user protocol, guaranteeing the delivery of messages, that *automatically* adapts to any new packet format that is obtained by applying a short, efficient function to packets from an earlier protocol.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.4 [**Performance of Systems**]: Fault tolerance, Reliability, availability, and serviceability

General Terms

Theory, Reliability

1. INTRODUCTION

Although the general public tends to regard computational infrastructure as appliances, the rapid growth and change of computational infrastructures invalidates this conception: software needs to be kept up-to-date. As the Internet grows, it grows less feasible to expect that *all* of the devices could be modified to cope with *any* change in the underlying protocol. And yet, the address space of the original protocol (IPv4) is nearing exhaustion, necessitating just such a change. Although a replacement protocol, IPv6, was established as a standard over a decade ago, the adoption rate has been modest at best. Many end-users are unaware of the issue, and those who are aware are generally reluctant to upgrade until incompatibility presents a problem. Furthermore, conversely, the obstacles to adoption of a new standard naturally instill a sense of conservatism in the development of new standards: even if a redesign of the protocol would improve some aspect of the network from users' perspectives, it stands little chance of being deployed unless it is compatible with the existing protocol.

The bottom line is that due to the scale of the Internet and the expectations or lack of awareness of users, the current model for updating computer infrastructure is inadequate. Motivated by this, Juba and Sudan [6] considered whether or not it would be possible for computers to adapt to changes in protocols automatically. They suggested that, by focusing on the *goal* of the communication – i.e., the functionalities provided – it might be possible to design communication protocols that support broad compatibility. Subsequent work by Goldreich et al. [4] developed a framework for such problems, and established roughly that whenever correct functioning can be locally verified, broad compatibility can be guaranteed. Furthermore, they show that guaranteeing correct functioning while supporting broad compatibility *requires* local verifiability.

Returning to our motivating example of the internet-layer packet protocol, we note that this protocol is designed to support the functionality of a (unreliable) channel. Unfortunately, Juba and Sudan [6] observed that this *specific* goal for communication cannot be guaranteed for broad classes of protocols, due to the inability of end-users to distinguish between incompatible encodings of distinct messages (an inability to locally verify correctness). So, unfortunately, we can't hope to design such a packet protocol with broad compatibility. Nevertheless, the observation does not rule out the possibility that for some *specific* higher-level functionality, end-users might still be able to adapt to changes in the lower-level packet protocol—e.g., specifically for a transport-layer protocol similar to TCP [2] to adapt to changes in the supporting internet-layer protocol, as we consider here.

Our main result in this work is the construction of a modified version of TCP that achieves essentially the same ends – implements a reliable channel – and can automatically adapt to "sufficiently small" changes in the internet-layer protocol, with the caveat that either the users must know each others' addresses or else there can only be two users. The main contribution in this work is in showing how the higher-level protocol can be modified to provide the necessary feedback to the lower-level protocol to enable adaptive behavior. A secondary (related) contribution is the definition of an appropriate class of "sufficiently small" changes for which it is possible to prove that the protocol adapts to changes in a reasonable amount of time: an issue with the techniques in prior works [4, 6] is that the overhead of the adaptive protocol (in terms of the running time or number of errors, drops, etc.) tends to grow exponentially with the size (e.g., in bits) of the unknown protocol. In this case, we bound the overhead in terms of the size of the *modification* to an earlier protocol, which we hope to be much more reasonable.

2. PROBLEM AND SOLUTION CONCEPT

We use the framework introduced by Goldreich et al. [4] to model communication without a fixed protocol. Essentially, this means that we fix a "goal for communication"

*Adapted from the author's Ph.D. thesis [5, Chapter 9].

†Supported by NSF Award CCF-0939370

PODC'11, June 6–8, 2011, San Jose, California, USA.
ACM 978-1-4503-0719-2/11/06.

capturing the functionality we wish to support, and a class of (network) protocols with respect to which we wish to exhibit compatibility. In this work, we consider communication from the perspective of a pair of end-users. The goal for our end-users is to realize a channel using a network connecting them: that is, each end-user receives as input an infinite sequence of messages (of some bounded maximum length N) and the goal is achieved if the users respectively produce their partner's input sequence as output. The network, modeling the Internet, is assumed to have a packet format for each of the users such that when a packet in the respective users' formats is sent to the network, the network (probably, eventually) forwards the packet to that user. We allow the network to drop packets (randomly with some fixed probability δ) and delay and/or reorder the packets adversarially up to some known maximum delay bound D; if there are too many packets addressed to one of the users in a given period (e.g., if both users together attempt to send more than D packets to one user) then the network is also allowed to drop new packets addressed to that user.

Now, we suppose that the network is chosen adversarially from a class of networks using different packet formats that we describe below. The problem is for the end-users to achieve their goal with the unknown packet format. Our solution concept is an algorithm for the end-users such that they correctly receive each others' messages when communicating across this network, no matter which packet format (from the class to be specified) is chosen. Naturally, the larger this class of protocols is, the more flexible the end-users are, but there are limits to what we can hope for. In particular, Goldreich et al. [4] show that in general there is an exponential lower bound on the number of rounds needed to achieve a goal in terms of the length of a protocol. Therefore, in order to achieve reasonable performance, it is necessary to restrict the class of protocols somehow. Our choice of restriction here is to assume that some protocol is known to the end-users, and that the network protocol is a "small modification" of this known protocol. Specifically, we assume that packets encoded in the old format may be translated to valid packets in the new format by some function described by a program of some a priori bounded length ℓ. Thus, although we may pay an exponential price in terms of the lengths of these modifications, we may still be able to adapt to gradual changes to the protocol over time without paying too severe a price.

The specific class of protocols we consider are packet network protocols that moreover compute the packet encoding in a single pass over the message, while using at most b bits of state (i.e., 2^b distinct states). It is interesting to consider such a weak and restricted class of protocols since we merely wish to guarantee "compatibility" with some relatively broad, explicit class of modifications. Our class of protocols is powerful enough to attach headers and compute checksums, and is therefore powerful enough to compute the encodings of most real packet network protocols.

3. THE ADAPTIVE END-USER PROTOCOL

Prior work by Goldreich et al. [4] implies that the ability to verify correct functioning is necessary and sufficient to construct solutions to our compatibility problem. Thus, the main idea is that the end-users can attach a MAC (or signature, e.g., as proposed by Gilbert et al. [3]) to their messages prior to encoding so that the recipient can verify when the message is decoded correctly; unconditional constructions of such MACs exist because the number of states b of the protocol is bounded, as is the size of the modification ℓ.[1] Thus, users can decode correctly (e.g., via trial-and-error). Given that the users can decode correctly, they can run a sliding-window scheme (following TCP [2]) so that they can confirm that a message was sent via an acknowledgement, and hence they can learn to encode messages as well.

The construction sketched above is sufficient to guarantee correctness, but because of the unreliability of the network, enumeration achieves poor performance—a protocol may fail due to a drop, and a naïve enumeration then cycles through the entire enumeration before returning to the "right" format. This performance issue can be circumvented by using an algorithm for the nonstochastic bandit problem [1] to select a candidate modification, rewarding the algorithm's choice whenever an acknowledgement is received, so the total reward received by the algorithm is a lower bound on the number of messages sent. (This *doesn't* reduce the initial overhead or improve the protocol search, but rather achieves a long-term transmission rate that approaches optimal.) The only difficulty is that the algorithm's choice needs to be evaluated before it makes a new choice, so the algorithm as stated can only handle a send window of size one. We circumvent this issue by using a number of copies of the algorithm equal to the (maximum) size of the send window, and invoking them in round-robin fashion. This increases the rate of failures due to bad choices by (only) the square root of the maximum size of the send window, which is independent of the total number of messages we attempt to send, so over time the algorithm's overall success rate still approaches the optimal rate of $1 - \delta$.[2]

4. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2003.

[2] V. G. Cerf and R. E. Kahn. A protocol for packet network intercommunication. *IEEE Trans. Comms.*, Com-22(5):637–648, 1974.

[3] E. N. Gilbert, F. J. MacWilliams, and N. J. A. Sloane. Codes which detect deception. *Bell Sys. Tech. J.*, 53:405–424, 1974.

[4] O. Goldreich, B. Juba, and M. Sudan. A theory of goal-oriented communication. Technical Report TR09-075, ECCC, 2009.

[5] B. Juba. *Universal Semantic Communication.* PhD thesis, MIT, 2010.

[6] B. Juba and M. Sudan. Universal semantic communication I. In *Proc. 40th STOC*, 2008.

[7] J. Kamp, A. Rao, S. Vadhan, and D. Zuckerman. Deterministic extractors for small-space sources. In *Proc. 38th STOC*, pages 691–700, 2006.

[1] Although we assume that the key for the MAC is known to both end-users, we can circumvent this assumption by using *deterministic extractors*, as constructed for the class of encodings we consider here by Kamp et al. [7], to exchange keys during a handshake (without knowing the encoding).

[2] Still, to allocate feedback correctly, we retransmit the messages one at a time. As long as the probability of a drop is sufficiently small relative to the size of the send window, though, this achieves reasonable performance.